A Utility-Based Approach to Bandwidth Allocation and Link Scheduling in Wireless Networks

A Utility-Based Approach to Bandwidth Allocation and Link Scheduling in Wireless Networks

Qicheng Ma, David C. Parkes, and Matt Welsh
School of Engineering and Applied Science
Harvard University
{ma,parkes,mdw}@eecs.harvard.edu

ABSTRACT

We study the problem of optimizing aggregate user utility in wireless ad-hoc networks under the constraints of wireless interference. We develop a market-oriented approach to bandwidth allocation with a tâtonnement process and demonstrate its ability to effectively price bottleneck resource. One novelty is that we choose to price "interference goods" to capture the externality imposed by one application's use of the network on other applications. In making progress we also propose a modification to the CSMA protocol that is robust enough to handle a non-schedulable bandwidth schedule. Experimental results on simulated network topologies show that the market-based approach has better scalability than alternate approximation methods and is much more efficient in terms of runtime.

1. INTRODUCTION

1.1 Motivation

Traditionally, bandwidth allocation and link scheduling in wireless ad-hoc networks are performed using either TDMA- or CSMA-based techniques [12]. The latter requires unnecessary overhead due to contention and collision, while the former is not adaptive enough to fit real-time usage patterns, and requires careful allocation of time slots. Recent research has proposed algorithms to schedule network usage in response to specific flow demands in order to optimize a total throughput objective [3, 4, 6, 8]. However, aggregate throughput does not faithfully reflect the ultimate true value of network usage since it fails to distinguish different needs of different users.

Consider the example of a sensor network deployed in hospital environment [14]. Different services such as patient tracking, doctor paging and vital sign monitoring all have differing priorities, and the value of the network traffic for each service is not a simple linear function of the bandwidth consumed by the service. Moreover, each service may be able to consume a varying amount of bandwidth to achieve different quality of service. For example, an electrocardiograph (ECG) can either report a high-bandwidth waveform or a low-bit-rate heart rate. The latter has less overall value but consumes considerably lower bandwidth.

These usage scenarios present the need for a resource scheduler that is aware of application- and user-specific *utility*, rather than mere bandwidth usage. Hence we propose to consider the optimization of aggregate application utility as the central goal of a new generation of resource schedulers.

We consider the following problem: a centralized scheduler is located at the base station and coordinates the scheduling of various application flows on a wireless multihop network. Each flow specifies a source and destination node in the network and a utility function that depends on the achieved bandwidth. Whenever a new flow is added or the profiles of existing flows are changed, the central scheduler re-runs its algorithm (possibly in an incremental fashion) in order to compute a schedule that specifies next-hop target and bandwidth consumption for each flow on each node.

We assume the central scheduler has global knowledge of the connectivity and interference characteristics of the networks and can notify each node of the schedule. A good scheduler will seek to *maximize aggregate utility* while scaling to large networks with many flows. The approach we take is market-oriented, where for now we use a centralized price-adjustment method coupled with distributed controllers for each application where the controllers report demand-sets given provisional prices. A tâtonnement process continues until an (approximate) equilibrium is reached.

For now we are not interested in markets for their ability to mitigate issues of self-interest. This said, one appeal of market-based methods is that they do suggest the possibility of being able to handle self-interest, e.g. when users may try to misrepresent the bandwidth requirements of an application. Furthermore, the assumption of a *centralized* (market) planner and global knowledge may be relaxed in future research, for instance through replication of the current market pricing and scheduling function across nodes in the network.

We first introduce a formal definition for the utility-based optimization problem, and introduce a linear-programming relaxation based on earlier work of Jain et al. [6]. This relaxation is interesting for two reasons: (a) it is scalable whereas the full problem is NP-complete; and (b) it leads to a novel market approach where *virtual goods* — representing interference structures — are priced. This pricing of virtual, interference-related goods is an important part of our solution. We propose a modification to the CSMA method for Medium Access Control (MAC) that is robust enough to handle as an input a non-schedulable flow vector, which is generated by the approximation. This is also used in our market approach which can be viewed as an alternate approximation to the full optimization problem. We present detailed experimental results for all three schemes: optimal, approximate and market.

1.2 Related Work

Previous work has looked at the characterization of network capacity and link scheduling problem in multi-hop net-

works. Some of this work explores the problem of optimizing total bandwidth or throughput in multi-hop network given flow demands [3, 4, 6, 8], mostly employing LP's to formulate the optimization problem or develop theoretical bounds. The work of Jain et al. [6] is important here because it formalizes the use of a conflict graph to model interference and provides a formal optimization model for a non utility-based version of our problem.

Radunovic et al. [11] argue that total throughput may not be the right objective to optimize and introduce a class of utility functions but adopt them as a proxy for fairness rather than seeking explicitly utilitarian solutions. Andrews et al. [1] consider the optimal utility problem in the situation of multiple user sharing one single radio medium. Yang and de Veciana [17] consider both user and network utility to form a dual optimization problem but do not consider a market-based approach.

Wellman's [15] seminal work on market-oriented programming has inspired this work. Wellman and colleagues applied the idea to a broad range of resource allocation problems such as network transportation and multi-commodity flow problems [16]. Despite the negative results in general equilibrium theory about the stability and convergence of the competitive market equilibrium [2, 13], the tâtonnement process tends to work quite well in practice in certain types of computational markets.

2. PRELIMINARIES

Our formulation is based on the model of connectivity graph and conflict graph developed by Jain et al. [6], augmented here with the notions of applications and utilities.

For a given wireless network with n nodes, the *connectivity graph* C is a *directed* graph defined by $\langle N, L\rangle$ where $N = \{1, 2, ..., n\}$ is the set of nodes representing wireless devices and $L = \{l_{ij} : \text{there exists a link from } i \text{ to } j\}$ is the set of *directed* links among them. Each link has its capacity, $Cap(l_{ij})$, which is the maximum achievable data rate *if the link $l_{i,j}$ is active during the entire time span T*, to be defined later.

A set of applications $A = \{1, 2, \ldots, m\}$ are competing to use the network. Each application $k \in A$ is defined by a tuple $\langle s_k, d_k, u_k\rangle$, which specifies a source node s_k, a destination node d_k and a utility function u_k defined over the bandwidth of a flow allocated to it from s_k to d_k. The minimum requirements for a utility function is that it passes through (0,0) and is non-decreasing. We discuss further restrictions to utility functions, such as concavity and piecewise-linearity, below.

A flow f is an assignment of bandwidth on each link in L to each application, with $f_{ij}^k \geq 0$ denoting the bandwidth on link l_{ij} assigned to application k. The following *Flow Conservation Constraints* must be satisfied for every application k:

$$\sum_{j:l_{ji}\in L} f_{ji}^k - \sum_{p:l_{ip}\in L} f_{ip}^k = 0, \quad \forall i \in N \setminus \{s_k, d_k\} \qquad (1)$$

Equation (1) states that for every application k and every non-terminal node i, the total inflow must equal the total outflow. The application's utility is $u_k(f^k)$, where $f^k = \sum_{j:l_{ji}\in L} f_{ji}^k - \sum_{j:l_{ij}\in L} f_{ij}^k$ for $i = d_k$ denotes the total flow allocation to application k. Moreover, we call the complete flow assignment for every application the *flow vector*, denoted $f^A = (f^1, ..., f^m)$.

To model interference, define the *conflict graph* of the network to be an *undirected* graph $F = \langle V_F, E_F\rangle$, whose vertices $V_F = L$ correspond to the links in the connectivity graph. An edge $\langle l_{ij}, l_{pq}\rangle \in E_F$ means that the two links interfere with each other and cannot be active simultaneously.[1] The conflict graph is defined between links rather than between nodes to allow general models of interference.[2]

The *time span* T is the index set of time slots in an *epoch*, which is a repeating time period over which link scheduling decisions are made. For every element in the set T, a decision has to be made as to which subset of the link L will be active and for which application. Hence, we define a *schedule* to be a function $S : A \times L \mapsto \{0, 1\}^T$ that defines for each application the subset of the the total time span T for which the application is active on a link. E.g., $S(k, l) = \{1, 3, 5\}$ means in time slots 1,3,5 link l will be active transmitting application k's data. We say a schedule S is *feasible* if the following *Schedulability Constraint* is satisfied:

$$\forall l_{ij} \in L, k_1, k_2 \in A : k_1 \neq k_2 \Rightarrow S(k_1, l_{ij}) \cap S(k_2, l_{ij}) = \phi \qquad (2)$$

$$\forall l_{ij}, l_{pq} \in L, k_1, k_2 \in A : \langle l_{ij}, l_{pq}\rangle \in E_F \Rightarrow S(k_1, l_{ij}) \cap S(k_2, l_{pq}) = \phi \qquad (3)$$

Equation (2) states that no two applications can occupy the same link at the same time, and (3) states that no two links that interfere with each other can be active at the same time.

We say that a schedule S *implements* a flow vector $f^A = (f^1, ..., f^m)$ iff S is feasible and the following is satisfied:

$$\forall i, j \in N, k \in A, f_{ij}^k \leq Cap(l_{ij}) \cdot \frac{\|S(k, l_{ij})\|}{\|T\|} \qquad (4)$$

where $\|.\|$ denotes the size of a set. This equation means that the achieved bandwidth of an application on a link is no more than the capacity of the link scaled by the proportion of time the application is scheduled to occupy that link. We say a flow vector f^A is feasible if it can be implemented by some feasible schedule S.

The *Optimal Utility Scheduling Problem* (OPT) is the following: given the connectivity graph C, the conflict graph F and a list of applications A, compute a feasible schedule S that maximizes the aggregate utility of all applications:

$$\max_{f^A, S} \sum_{k\in A} u_k(f^k) \qquad (5)$$

$$\text{s.t. } f^A \text{ and } S \text{ satisfy (1)-(4)}$$

The interpretation of the time span T depends on the MAC protocol used at the link layer. In a TDMA-based scheme, T is a discrete set of time slots within an epoch, where an epoch corresponds to a superframe of several time slots on a repeating schedule. In a CSMA protocol, we could interpret T as an arbitrary set of time slots in which $|T|$ represents the fraction of slots in each epoch allocated to the node. T can of course be generalized to correspond to other ways of scheduling the link. For example, in a FDMA

[1]Note that we do not draw an edge from a vertex (link) to itself in the interference graph, but we do draw an edge between a pair of reverse links l_{ij} and l_{ji}.

[2]We make the simplifying assumption that there is no partial interference but rather that interference is a boolean concept. See Padhye et al. [10] for a discussion of this issue.

scheme, T could correspond to the set of frequencies, rather than time slots.

The *utility function* is a map from overall bandwidth to the application's value for being able to achieve the given bandwidth level. There are several properties of utility functions considered in our model: First we require that it pass through (0,0) by convention. Second it must be non-decreasing (otherwise the application may just step down to operate at a lower bandwidth than it is allocated). Finally we restrict it to be in the class of piecewise linear functions, including ones with discontinuous jumps. This restriction offers two advantages. First, such functions have compact representations and yet are able to approximate any reasonable functions. One way to specify a piecewise-linear function is to give a list of breakpoints (x_i, y_i), and two extra slopes *preSlope* and *postSlope* before the first point and after the last point. Second, piecewise linear functions are suitable for integer linear programs.

Although we do not strictly require utility functions to be concave, typical applications tend to have diminishing marginal returns on bandwidth and thus have concave utility functions. Figure 1 shows two examples of piecewise linear concave utility functions. The first utility function (0,0)-(1,10)-(2,15) represents an application with two modes of operations and a decreasing marginal utility of each additional unit of bandwidth. We will use this typical utility function extensively in later experimental sections.

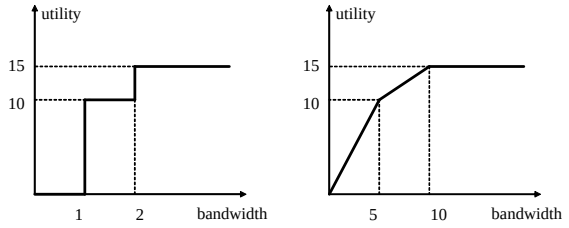

Figure 1: **Examples of piecewise-linear concave utility functions with discrete (left) or continuous (right) bandwidth**.

2.1 Complexity and Approximations

Jain et al. [6] have established that it is NP-hard to compute the throughput-maximizing schedule in the presence of interference, and moreover it is NP-hard to produce an approximation within a constant ratio of the optimal solution. Since their formulation of the maximum throughput scheduling problem is a special case of Opt with continuous time span and identity utility function, Opt is also NP-hard. This said, in our experimental work we formulate and solve Opt as a mixed-integer program (MIP) and use CPLEX[3] where a solution is available in reasonable time. This provides one of the benchmarks for the market system.

2.1.1 An LP relaxation

Following the ideas in Jain et al. [6], we can relax the MIP formulation and consider instead a linear-programming (LP) relaxation. This LP relaxation can be strengthened through the introduction of additional non-violated constraints, namely clique and odd-hole constraints.

The *clique constraint* is the following: A clique Q in the conflict graph is a subset of vertices with edges between every pair of them, i.e. a set of links that mutually interfere with each other. It is clear that at any given time period only one link in a clique can be active, and consequently the total number of active time periods of all links in a clique must be less than the total number of time periods. Formally, for a clique Q:

$$\sum_{l_{ij} \in Q} \sum_{k \in A} \left\lceil f_{ij}^k \frac{\|T\|}{Cap(l_{ij})} \right\rceil \leq \|T\| \tag{6}$$

The scaling factor $\frac{\|T\|}{Cap(l_{ij})}$ translates the bandwidth assignment to the number of time periods used. It suffices to consider maximal cliques because a constraint for a non-maximal clique Q_1 is subsumed by the constraint for a maximal clique Q_2 if $Q_1 \subset Q_2$.

Similarly, the *odd-hole constraint* is derived from an odd-hole H, which is a cycle of odd number of links in the conflict graph. Since at any given time period at most half ($\frac{\|H\|}{2}$) of them can be active, the aggregate number of active time periods for all links in an hole over the entire epoch must be smaller than $\left\lfloor \frac{\|H\|}{2} \right\rfloor \cdot \|T\|$. Odd-holes offer non-trivial information compared to even-holes because of the rounding operator. Formally, for an odd-hole, H:

$$\sum_{l_{ij} \in H} \sum_{k \in A} \left\lceil f_{ij}^k \frac{\|T\|}{Cap(l_{ij})} \right\rceil \leq \left\lfloor \frac{\|H\|}{2} \right\rfloor \cdot \|T\| \tag{7}$$

Notice that the use of the floor and ceiling function makes the constraint tighter and leads to a more refined upper-bound. However, because the number of maximal cliques and odd-holes are in general exponential in the number of nodes we cannot possibly add all clique constraints and all odd-hole constraints. Instead, this suggests an approximation algorithm in which a *random* (as many as feasible given computational constraints) set of cliques and odd-holes constraints are introduced into the LP before it is solved.

Still, the difficulty with this approach is that the output is a flow allocation which may not be schedulable [6]. Here, we handle this through combining this LP-relaxation with a novel CSMA-like scheme to (approximately) implement this allocation in best-effort while ensuring schedulability. The solution of the LP-relaxation becomes the upper-bound flow-vector, or the "Optimal Rate Limiter," for the modified CSMA scheme. Together, we refer to this combination approach as the Orl-Csma scheduler.

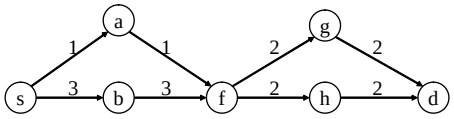

Figure 2: **Examples of a multi-path flow-vector**. *Node f has multiple outflows and will forward incoming packages to g or h according to a dispatch schedule.*

2.1.2 Modified CSMA

Every node will transmit data only in units that are equivalent to the time length of one round in the TDMA schedule, which we call a "package".[4] At the beginning of each round, there is a short contention period in which nodes perform

[3] `http://www.ilog.com`

[4] A package could consist of multiple packets at the PHY layer.

Type of good	Members	Supplied Quantities
Link Pair (L)	$L = \{l_{ij}, l_{ji}\}$	$\|T\|$
Clique (Q)	$\forall l_{ij}, l_{pq} \in Q : \langle l_{ij}, l_{pq} \rangle \in E_F$	$\|T\|$
Odd-Hole (H)	$H = \{l_{i_0 j_0}, l_{i_1 j_1}, ..., l_{i_{h-1} j_{h-1}} : \langle l_{i_k j_k}, l_{i_{k \oplus 1} j_{k \oplus 1}} \rangle \in E_F\}$	$\left\lfloor \frac{\|H\|}{2} \right\rfloor \cdot \|T\|$

Table 1: Types of goods and their supplied quantities in MARKET

conventional CSMA-style carrier sensing and backoff in case of collision. We assume that the contention period is short compared to the round length, and hence the beginnings and ends of all transmissions are aligned to round boundaries.

The source node of a flow will issue packages addressed to its neighbors at a rate and fashion specified in the upper-bound flow-vector. In the example upper-bound flow-vector shown in Figure 2, source node s will issue 1 package to a and 3 packages to b during one epoch. A forwarding node will attempt to retransmit each received package at a random time in the next contention period to the next downstream neighbor according to the flow-vector.

A forwarding node f with multiple outflows (and perhaps multiple inflows) services each outflow according to a *dispatch schedule*. The dispatch schedule is assigned a random permutation of the node's outflows on each epoch. In this way, we avoid biasing the delivery latency of flows that would result from using a fixed transmission schedule.

3. THE MARKET-BASED APPROACH

In this section we will present a market-oriented approach to approximate the above LP-relaxation approach. Here too we make use of the modified CSMA scheduler, and the combination of the market allocation method with the use of the best-effort CSMA scheduler becomes the MARKET-CSMA algorithm. We adopt *tâtonnement* as an approach to converge towards a market equilibrium. An overview of the MARKET-CSMA algorithm is provided in Figure 3.

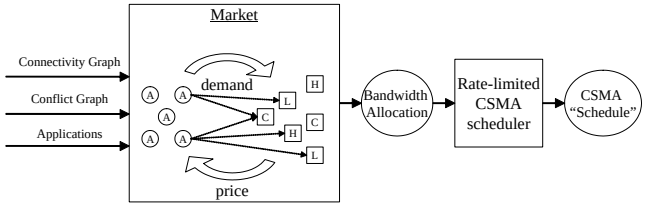

Figure 3: MARKET **protocol overview**. *First a tâtonnement process is used to iterate between price-updates and best-response from applications. Then, when approximate convergence is achieved the equilibrium allocation is passed on to the best-effort CSMA scheduler as the rate limiter.*

In light of the constraints presented in the ORL problem and its LP-relaxation, we price links, cliques and odd-holes in the market to capture the constrained resources. Formally, a **good** in our market is defined as *a set of links*, which we will call an *interference group*. A good indexed by the interference group g is a license or permission to use *any one* of the links in g for one period of time. Conversely, in order to be able to use a physical link l in the final allocation, an application has to purchase *all* of the goods that correspond to an interference group containing l. In particular, we consider three types of goods, namely: *link pair*, *clique* and *odd-hole* goods. The supplied quantities of each type of good are as shown in Table 1.

The introduction of cliques and odd-holes as virtual goods in addition to the physical link-pair goods in order to capture the effect of interference appears to be a novel contribution. In our experiments we find that these interference goods (clique goods in particular) are the most demanded goods in the market.

Suppose there are a total of L physical links $\{l_1, ..., l_L\}$ and G goods $\{g_1, ..., g_G\}$, we use the G-dimensional vectors $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{x}$ to denote the *price vector*, *supply vector* and *demand vector* for the goods, respectively. On the other hand, because an application works naturally by first selecting a set of physical links that it wants to use and then procuring all the goods required to operate those links, we will also define the *effective link price vector* $\mathbf{p}^L$ and the *link demand vector* $\mathbf{x}^L$ as an alternative representation of the prices and demands. The effective link price p_i^L for each link l_i is the sum of prices of all goods that the link is a member of:

$$p_i^L = \sum_{1 \le j \le G : l_i \in g_j} p_j \tag{8}$$

In response to these effective link prices, each application $k \in A$ states its demand on each link it wants to use (in order to form a flow), which we will call the *application-level link demand vector* $\mathbf{x}_k^L$. The component x_{ki}^L denotes the quantity of physical link l_i that application k demands. The sum of $\mathbf{x}_k^L$ across all applications k becomes the *aggregate link demand vector*, denoted $\mathbf{x}^L$. This maps to a demand on goods g_j as:

$$x_j = \sum_{1 \le i \le L : l_i \in g_j} x_i^L \tag{9}$$

The MARKET protocol is defined as follows:

1. $\{g_1, ..., g_G\} \leftarrow CHOOSE\_GOODS$.
2. $t \leftarrow 0$, $\mathbf{p}(0) \leftarrow \mathbf{0}, H(0) \leftarrow \phi$.
3. Price vector $\mathbf{p}(t)$ is announced.
4. Each application $k \in A$ responds to $\mathbf{p}(t)$ by doing:
   (a) Translate good-price $\mathbf{p}(t)$ to effective link price $\mathbf{p}^L(t)$.
   (b) Compute its link demand vector $\mathbf{x}_k^L(t)$ $= APP\_DEMAND(k, \mathbf{p}^L(t))$.
   (c) Translate link demand $\mathbf{x}_k^L(t)$ to good demand $\mathbf{x}_k(t)$.
   (d) Submit quantity demanded, $x_{kj}(t)$, for each good g_j to the auctioneer s_j for good g_j
5. Each auctioneer s_j sums over all bids from applications to get the aggregate demand $x_j(t)$ for good g_j.
6. If $CONVERGE?(H(t), \mathbf{p}(t), \mathbf{x}(t))$ output $\mathbf{x}_k^L(t)$ as the link allocation for each application k.

7. Otherwise *one* auctioneer s_j is selected at random from the set for which $x_j(t) \neq q_j$. This auctioneer updates the price and all other prices remain unchanged: $\mathbf{p}(t+1) \leftarrow UPDATE\_PRICE(j, \mathbf{p}(t), \mathbf{x}(t))$.

8. $H(t+1) \leftarrow H(t) \cup \{\mathbf{p}(t), \mathbf{x}(t)\}$, $t \leftarrow t+1$, go to Step 3.

Note that $H(t)$ is the history of price vectors and demand vectors prior to time t. Upon termination of the protocol, the equilibrium link demand $\mathbf{x}_k^L$ for each application $k \in A$ becomes the final *link allocation*, which specifies how many of each link every application is allowed to use during the entire epoch of $\|T\|$ rounds. Under the simplifying assumption that $\frac{Cap(l_i)}{\|T\|} = 1$ for all links, $\mathbf{x}^L$ is also conveniently the flow vector (bandwidth allocation), which can be passed on to the next stage CSMA scheduler as the rate-limiter.

There are four main modules within the market. The initial *CHOOSE_GOODS* function will always include all link-pairs, together with some subset of the possible hole and clique goods.

APP_DEMAND: Given the price vector $\mathbf{p}^L$, the sub-problem facing each application k is to find the optimal link demand in order to form a flow that gives the maximum net utility:

$$\mathbf{x}_k^L(t) \in \arg\max_{\mathbf{x}^L \in X^L} u_k(bw(\mathbf{x}^L)) - \mathbf{x}^L \cdot \mathbf{p}^L(t), \tag{10}$$

where $bw(\mathbf{x}^L)$ denotes the bandwidth of $\mathbf{x}^L$ (treated as a flow vector), and the dot-product computes the total price of the link allocation $\mathbf{x}^L$ the application has to pay under the current link price vector. The dual meanings of $\mathbf{x}^L$ as both the link demand and the corresponding flow vector are due to the assumption that $Cap(l_i) = \|T\|$ for all links.[5]

Obviously the set of all possible link demands that the application is allowed to choose from (X^L) should be limited by the flow conservation constraints. However, we do not add the constraints that an application cannot demand more than the total supplied quantities of goods, since adding these supply constraints makes the sub-problem too complex.

Without considering the supply constraints, the single-application optimal flow problem reduces to a classic shortest path problem with the effective link prices $\mathbf{p}^L(t)$ as the distance metric. Each application will simply compute the shortest distance (minimum cost in this case) path between its source and destination nodes, and will request multiple units of the links along that path. The number of units requested will be chosen to maximize net utility, and in the case of concave utility function, the application will keep increasing this number until the cost is higher than its marginal utility.

UPDATE_PRICE: For price tâtonnement we use the following *Simple Reinforcement / Negative-Feedback Update Rule:* The auctioneer j selected in step (7.) will simply push the price up or down by a small increment δ depending on whether a good g_j is over-demanded or under-demanded. That is, $p_j(t+1) \leftarrow p_j(t) + \delta$ if $x_j(t) > q_j$ and $p_j(t+1) \leftarrow p_j(t) - \delta$ if $x_j(t) < q_j$.[6]

[5] In general, an arbitrary flow vector $\mathbf{f}$ can be translated to a link demand by scaling the component for link l_i by a factor of $\frac{\|T\|}{Cap(l_i)}$ and rounding up to the next integer.

[6] Proportional updates can also be used ($p_j(t+1) \leftarrow p_j(t) + \delta(x_j(t) - q_j)$ with smaller δ). However, in our experiments a fixed increment tends to lead to less fluctuation, probably because proportional changes tend to overshoot the efficient prices due to the failure to expect discontinuous changes of the demands around the efficient prices.

CONVERGE? It is well known, e.g. Scarf [13] that market dynamics may reach a cycle without converging to the unique competitive equilibrium. In a discrete price and demand space in our setting, we expect the market to fluctuate among a limited number of states after a certain number of initial iterations. Indeed this cycling behavior is frequently observed in our experiments. When the market falls into a trap and cycles through only a limited number of states, we say that it reaches *pseudo-convergence*, and the set of states that it cycles through form a dynamic equilibrium.

The following simple algorithm is used to detect convergence of prices and demands. Consider the magnitude of the first order differences of the price vector and demand vector, if they tend to zero, the system reaches ideal convergence; if they stabilize around non-zero values, the system is fluctuating between a relatively stable set of states.

The *Exponentially Weighted Moving Average* (EWMA) with parameter α for any scalar or vector variable x is:

$$E_\alpha^*[x](t) = \alpha E_\alpha^*[x](t-1) + (1-\alpha)x(t) \tag{11}$$

Using this, we define the *average price vector* at time t to be an exponentially weighted moving average of historical price vectors:

$$\tilde{\mathbf{p}}(t) = E_\alpha^*[\mathbf{p}](t) = \alpha\tilde{\mathbf{p}}(t-1) + (1-\alpha)\mathbf{p}(t) \tag{12}$$

The scalar *first order difference* of the price vector is defined as the L-2 norm of the difference between the current price vector and the average price vector:

$$\Delta_{\mathbf{p}}(t) = \|\mathbf{p}(t) - \tilde{\mathbf{p}}(t)\| \tag{13}$$

Because this is still very volatile, we apply another EWMA filter on it to obtain the *average price vector difference*:

$$\tilde{\Delta}_{\mathbf{p}}(t) = E_\beta^*[\Delta_{\mathbf{p}}](t) \tag{14}$$

Finally the algorithm detects a pseudo-convergence in price vector when $\tilde{\Delta}_{\mathbf{p}}(t)$ stabilizes, i.e. its "moving standard deviation" is less than a certain percent of its moving average.

$$\frac{Std_\gamma^*[\tilde{\Delta}_{\mathbf{p}}](t)}{E_\gamma^*[\tilde{\Delta}_{\mathbf{p}}](t)} \leq \epsilon \tag{15}$$

where $Std*_\gamma$ is similar to the traditional standard deviation but with the expectation operators replaced by E_γ^*:

$$Std_\gamma^*[x](t) = \sqrt{E_\gamma^*[x^2](t) - (E_\gamma^*[x](t))^2} \tag{16}$$

Similar quantities are defined for the demand vector $\mathbf{x}(t)$ and the algorithm determines that the market reaches a pseudo-convergence when condition (15) and its counterpart for demand vector are both met.

The weights $(\alpha, \beta, \gamma, \epsilon)$ in the range (0,1) are the parameters of the detection algorithm and determine how sensitive it is to fluctuations and final stabilization. In general, the smaller they are, the long it takes for the algorithm to detect convergence. They are chosen empirically to be (0.90, 0.95, 0.95, 0.05) in the final implementation, which is able to detect pseudo-convergence in all test cases. Although the three rounds of EWMA smoothing filters seem excessive,

they are usually necessary due to the high fluctuation of the price and demand vectors.

In Figure 4 we plotted the three quantities $\Delta_{\mathbf{p}}(t)$, $\tilde{\Delta}_{\mathbf{p}}(t)$ and $Std^*_{\gamma}\tilde{\Delta}_{\mathbf{p}}(t)$. The corresponding versions of the demand vector are shown in Figure 5. Notice the contrasting behavior of the price and demand vector difference at around iteration 1000: while the price vector difference drops significantly and stabilizes at a lower level, the demand vector difference rises and stabilizes at a higher level.

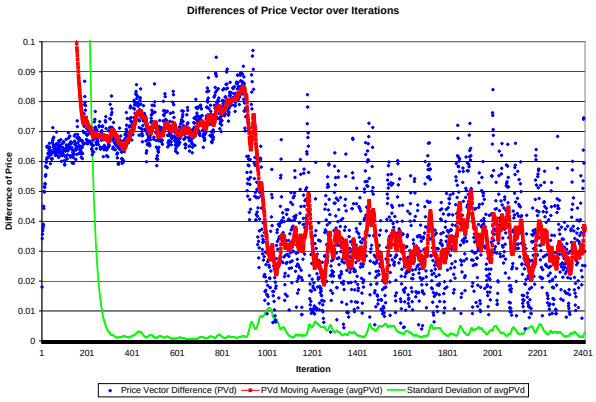

Figure 4: **Difference of Price Vector over Iterations**.

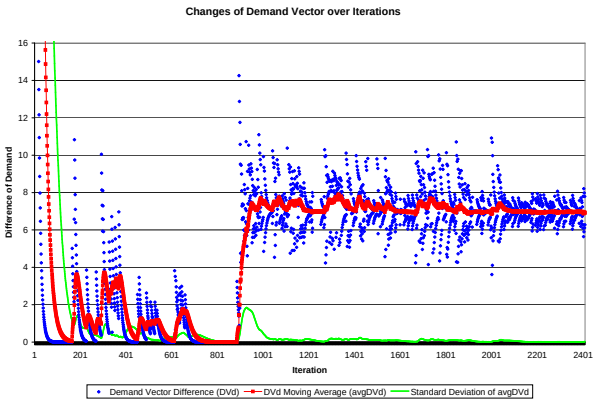

Figure 5: **Difference of Demand Vector over Iterations**.

When the MARKET algorithm detects pseudo-convergence, it simply outputs the last market state to the next stage CSMA scheduler.

4. EXPERIMENTAL RESULTS

We compare the performance of four algorithms: OPT, ORL-CSMA, MARKET-CSMA, and NAIVE-CSMA. NAIVE-CSMA is a simple CSMA-based protocol in which nodes simply contend for the radio whenever they have data to send; no bandwidth allocation is performed. Recall that OPT is the MIP formulation of the optimal utility scheduling problem and explicitly constructs a feasible schedule to maximize the aggregate utility of all applications.

We make use of two different interference models. In the *Level-0* model, two links sharing an endpoint will interfere with each other. In the *Level-1* model, in addition to Level-0 interference, two links will interfere if the receiver endpoint of one link is in range of the sender of the other link. This model is more realistic in that links without an endpoint in common, but with nearby endpoints, can interfere.

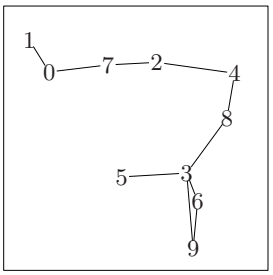

Figure 6: **A randomly generated network to carry out case study**. *10 nodes are placed uniformly at random in the unit square, with symmetric directional links connecting every pair of nodes within distance* 0.3.

To characterize the performance of the four algorithms as demand increases we first perform a case study on a specific instance of a randomly generated network shown in Figure 6. We assume links have identical capacities of 10, which is also the number of time periods. We generate a list of homogeneous applications with random sources and destinations and our typical (0,0)-(1,10)-(2,15) utility function in Figure 1. We include them, in order, to form a sequence of test cases to be scheduled by each algorithm. We assume a Level-0 interference model. For *CHOOSE_GOODS* we include all link-pair goods, all clique goods but no hole goods.

In Figure 7 we report the running time, average link usage, total and average bandwidth and utility on all test runs. In addition, we also show the performance that would be achieved if the solution to the LP-relaxation ORL was schedulable. This is labeled "csma-ub" in the plots while the ORL-CSMA method is labeled "csma-sim."[7] Since data points are taken more sparsely as the number of applications increases, the x-axis has been compressed by a factor of 4 in the right half of each graph for better visual presentation. Some important observations from these results are:

- **All three algorithms outperform** NAIVE-CSMA **significantly in both bandwidth and utility.** NAIVE-CSMA suffers from overwhelming contention in high demand and has degrading performance.
- **Run-time complexity** OPT$\succ$ORL-CSMA$\succ$MARKET-CSMA**.** In terms of running time, OPT grows worse than exponentially in the number of applications (note the time axis is in log-scale), ORL-CSMA grows approximately exponentially and MARKET-CSMA is almost flat. There is an initial jump from 2 to 3 applications of MARKET-CSMA because prior to 3 applications, the demand is so low even at all zero prices that the market terminates immediately. The "running time" of NAIVE-CSMA is not plotted because it does not require prior computation.
- **Link Usage very similar, with** NAIVE-CSMA **slightly worse.** NAIVE-CSMA used the most link time

[7]Note that the bandwidth line of csma-ub is not necessarily a strict upper-bound because the first stage ORL-CSMA optimizes for utility rather than bandwidth.

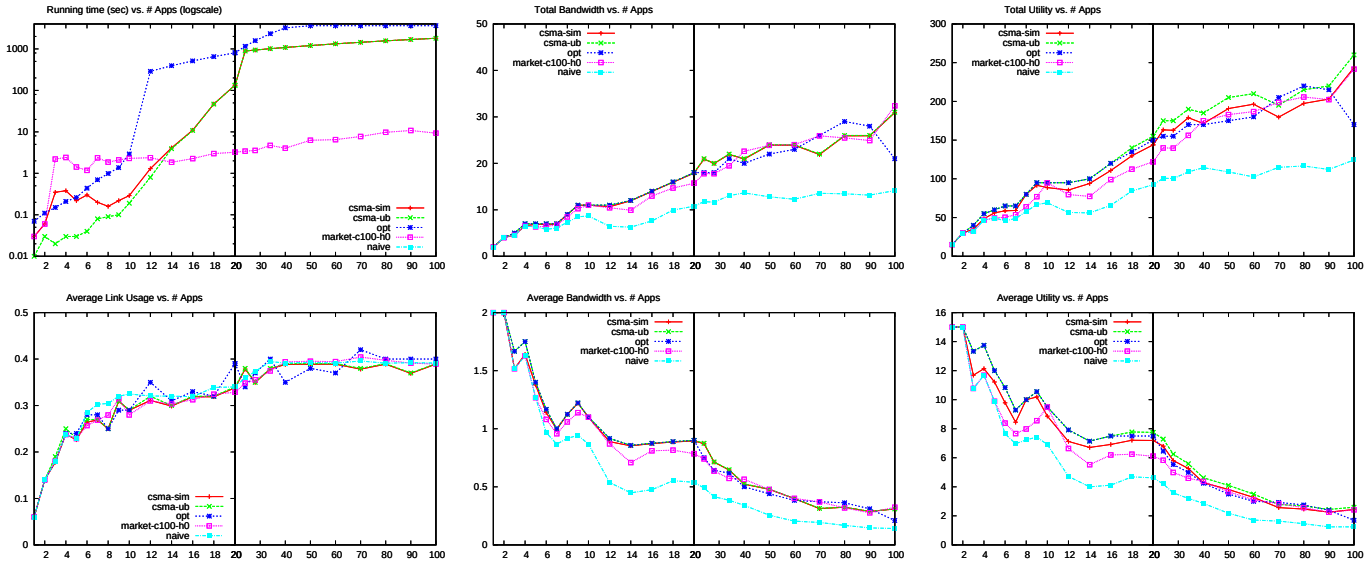

Figure 7: **Running Time, Link Usage, Total and Average Bandwidth and Utility of** Opt, Orl-Csma **("csma-sim"),** Market-Csma **and** Naive-Csma**. The performance that would be achieved if the solution to the LP-relaxation** Orl **was schedulable is also shown ("csma-ub").** *Note that the right half of each graph is compressed by a factor of 4 along the x-axis.*

since it tends to waste a lot of bandwidth in delivering messages that will get dropped later. The absolute upper-bound on link usage is 0.5 because at most 5 links can be active simultaneously under the Level-0 interference model, as shown in figure 6.

- **Bandwidth** Opt≈Orl-Csma≈Market-Csma**.** Although the three algorithms are not geared towards bandwidth maximization, it makes sense that bandwidth must be roughly maximized in order to achieve maximum utility. The Bandwidth characteristics of the three algorithms are virtually indistinguishable, with Market-Csma slightly lower than Opt and Orl-Csma before they hit computational limits.

- **Utility** Opt≻Orl-Csma≻Market-Csma **(but close!).** Despite similar usage of bandwidth resources, the utility levels show real differences in performance. In low demand when computational constraint is not yet harsh for Opt,Orl-Csma tracks Opt very closely (about 95%) while Market only tracks about 80%-90% of Opt's utility.

- **Computationally constrained** Opt **underperforms** Orl-Csma **and** Market-Csma **in high demand.** After Opt hits the computational bound (>20 apps), Orl-Csma begins to outperform Optin utility. Similarly after Orl-Csma hits the computational bound in extreme high demand (>70 apps) Market-Csma catches up. Because the time limit we set is extremely generous (30 minutes), we expect that Market would be the most scalable algorithm in practical settings.

For our second set of experiments we assess the performance for a distribution over different network topologies, interference models and utility functions. In each test case, a network is generated with random placement of a random number (5-15) of nodes, and the nominal range l chosen from $(0.3, 0.4)$. A random number (5-20) of applications with random sources, destinations and randomly generated utility functions will enter. A utility function is generated by choosing a random range (1-5) of bandwidth and then choosing random utility value (0-20) at each bandwidth level, guaranteeing non-decreasing but not concavity. We fix the number of rounds at 10 and use the Level-0 or Level-1 interference model at random. Each test case will run for at most 10 minutes by each algorithm.

In addition to average bandwidth, average utility and link usage, we also look at a fairness metric [7] on bandwidth and utility vectors. The fairness index for a vector $\mathbf{x}$ with dimension n is defined as:

$$fairness(\mathbf{x}) = \frac{(\sum x_i)^2}{n \cdot \sum x_i^2} \tag{17}$$

A value closer to 1 means more fair. Table 2 summarizes the performance metrics. All numbers are relative to Naive-Csma as a baseline.

Algorithm	Opt	Orl-Csma	Market-Csma
Utility	1.20x	1.22x	1.13x
Bandwidth	0.86x	0.95x	0.89x
Utility Fairness	0.89x	0.98x	0.99x
Bandwidth Fairness	0.76x	0.84x	0.88x
Link Usage	0.97x	1.04x	0.89x

Table 2: Performance Metrics of Opt, Orl-Csma and Market-Csma, relative to Naive-Csma

In terms of efficiency, all three algorithms use less aggregate throughput (and lower link usage except Orl-Csma) to achieve higher aggregate utility compared to Naive-Csma. Market-Csma tracks about 95% utility of Opt. Note that Orl-Csma seems to outperform Opt on average in this ex-

Time Limit	OPT	ORL-CSMA	MARKET-CSMA
10 min	52%/90%	100%/100%	100%/100%
5 sec	?/37%	?/96%	100%/100%

Table 3: Performance Metrics of OPT, ORL-CSMA and MARKET-CSMA

periment due to the moderate computational constraints. In terms of fairness, OPT is much poorer than the two CSMA-based algorithms, probably due to the fact that OPT computes and fixes a TDMA schedule once and for all, while the CSMA-brand algorithms have the second stage best-effort scheduler that contributes to a mixing and smoothing of bandwidth and utility across applications.

The time limit of 10 minutes imposes moderate computational constraints that elevates the performance of ORL-CSMA compared to OPT. We can also expect that under harsh computational constraints (such as in an online environment, e.g. 5 seconds) MARKET-CSMA will be the best choice. To stress this point, we illustrate the solution quality of the three algorithms under various time constraints in Table 3. The first number is the percentage of test runs in which the algorithm returns an *optimal* solution (meaning finding the optimal solution in MIP stage, or finding market convergence). The second number is the percentage of test runs in which the algorithm returns a solution *at all*, possibly suboptimal.

5. CONCLUSIONS AND FUTURE WORK

It is conceivable that the MARKET allocation coupled with CSMA scheduler could become an online realtime scheduling protocol. The demand response in the market could be continuously fed to the second stage CSMA scheduler without waiting to settle down on a particular stable state. A distributed and decentralized implementation can be achieved by delegating each auctioneer onto some individual nodes, and propagating the price information and bid information by piggy-backing. A real implementation would also need to consider the consequences of currency allocation policies (if the currency is virtual) (see for example [5]), as well as to specify utility functions appropriate for different applications [17, e.g.]. Another next step is to implement and deploy our algorithms on real-life wireless network test beds such as the MoteLab [9]. Further complication may arise due to realistic wireless behaviors such as collision and partial interference that's not captured by our model or simulator. Future research can also implement our algorithms in TinyOS and measure relevant performance metrics in real-life experiments.

6. REFERENCES

[1] M. Andrews, L. Qian, and A. Stolyar. Optimal utility based multi-user throughput allocation subject to throughput constraints.

[2] K. J. Arrow and L. Hurwicz. On the stability of the competitive equilibrium i. *Econometrica*, 26:no. 4, 522–552, 1958.

[3] R. Bhatia and L. E. Li. Characterizing achievable multicast rates in multi-hop wireless networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 133–144, New York, NY, USA, 2005. ACM Press.

[4] B. E. Hajek and G. H. Sasaki. Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, 34(5):910–917, 1988.

[5] D. Irwin, J. Chase, L. Grit, and A. Yumerefendi. Self-recharging virtual currency. In *Third Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[6] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 66–80, New York, NY, USA, 2003. ACM Press.

[7] R. Jain, A. Durresi, and G. Babic. Throughput fairness index: An explanation. In *ATM Forum*, pages 99–0045, 1999.

[8] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Algorithmic aspects of capacity in wireless networks. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 133–144, New York, NY, USA, 2005. ACM Press.

[9] motelab. Harvard sensor network testbed. web. http://motelab.eecs.harvard.edu.

[10] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *IMC '05: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2005. ACM Press.

[11] B. Radunovic and J.-Y. L. Boudec. Rate performance objectives of multihop wireless networks. *IEEE Trans. Mob. Comput.*, 3(4):334–349, 2004.

[12] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wirel. Netw.*, 5(2):81–94, 1999.

[13] H. E. Scarf. Some examples of global instability of the competitive equilibrium. Cowles Foundation Discussion Papers 79, Cowles Foundation, Yale University, 1959. available at http://ideas.repec.org/p/cwl/cwldpp/79.html.

[14] V. Shnayder, B. rong Chen, K. Lorincz, T. R. F. F. Jones, and M. Welsh. Sensor networks for medical care. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 314–314, New York, NY, USA, 2005. ACM Press.

[15] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[16] M. P. Wellman. Market-oriented programming: Some early lessons. In S. H. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*, chapter 4, pages 74–95. World Scientific, 1996.

[17] S. J. Yang and G. de Veciana. Enhancing both network and user performance for networks supporting best effort traffic. *IEEE/ACM Trans. Netw.*, 12(2):349–360, 2004.